TOP 25 SQL Q&A

Follow Instagram
@Career_with_kavitha

Top 25 SQL Q&A to clear 95% of Interviews

💡 Medium Level:

1️⃣ Find the Second Highest Salary in an Employee Table:

```sql
SELECT MAX(salary) AS second_highest_salary
FROM employees
WHERE salary < (SELECT MAX(salary) FROM employees);
```

2️⃣ Fetch Employees Whose Names Contain the Letter "a" Exactly Twice:

```sql
SELECT name
FROM employees
WHERE name LIKE '%a%a%' AND name NOT LIKE '%a%a%a%';
```

3️⃣ Retrieve Only Duplicate Records:

```sql
SELECT column_name, COUNT(*)
FROM table_name
GROUP BY column_name
HAVING COUNT(*) > 1;
```

Career

4️⃣ Calculate the Running Total of Sales by Date:

```sql
SELECT sales_date,
       SUM(sales_amount) OVER (ORDER BY sales_date) AS running_total
FROM sales;
```

5️⃣ Find Employees Earning Above Average Salary in Their Department:

```sql
SELECT emp_id, emp_name, department_id, salary
FROM employees
WHERE salary > (
    SELECT AVG(salary)
    FROM employees AS e
    WHERE e.department_id = employees.department_id
);
```

6️⃣ Find the Most Frequently Occurring Value in a Column:

```sql
SELECT column_name, COUNT(column_name) AS frequency
FROM table_name
GROUP BY column_name
ORDER BY frequency DESC
LIMIT 1;
```

Career_

7️⃣ Fetch Records Within the Last 7 Days:

```sql
SELECT *
FROM table_name
WHERE date_column >= CURDATE() - INTERVAL 7 DAY;
```

8️⃣ Count How Many Employees Share the Same Salary:

```sql
SELECT salary, COUNT(emp_id) AS employee_count
FROM employees
GROUP BY salary
HAVING employee_count > 1;
```

9️⃣ Fetch Top 3 Records for Each Group:

```sql
WITH ranked_data AS (
    SELECT column_name,
           RANK() OVER (PARTITION BY group_column ORDER BY value_column DESC) AS rank
    FROM table_name
)
SELECT *
FROM ranked_data
WHERE rank <= 3;
```

Career

🔟 Retrieve Products That Were Never Sold:

```sql
SELECT p.product_id, p.product_name
FROM products p
LEFT JOIN sales s ON p.product_id = s.product_id
WHERE s.product_id IS NULL;
```

💡 Challenging Level:

1️⃣ Retrieve Customers Who Made Their First Purchase in the Last 6 Months:

```sql
SELECT customer_id, MIN(purchase_date) AS first_purchase_date
FROM sales
GROUP BY customer_id
HAVING first_purchase_date >= CURDATE() - INTERVAL 6 MONTH;
```

Career_with

2 Pivot a Table to Convert Rows into Columns:

```sql
SELECT
    employee_id,
    MAX(CASE WHEN month = 'January' THEN salary END) AS january_salary,
    MAX(CASE WHEN month = 'February' THEN salary END) AS february_salary
FROM salary_data
GROUP BY employee_id;
```

3 Calculate Percentage Change in Sales Month-over-Month:

```sql
WITH sales_cte AS (
    SELECT sales_date,
           SUM(sales_amount) AS total_sales
    FROM sales
    GROUP BY MONTH(sales_date)
)
SELECT sales_date,
       (total_sales - LAG(total_sales) OVER (ORDER BY sales_date)) * 100 / LAG(total_sales
FROM sales_cte;
```

Career_Wi

4 Find the Median Salary of Employees:

sql

Copy code

```sql
SELECT AVG(salary) AS median_salary
FROM (
    SELECT salary
    FROM employees
    ORDER BY salary
    LIMIT 2 OFFSET (SELECT FLOOR(COUNT(*)/2) FROM employees)
) AS temp;
```

5 Users Logged In for 3 Consecutive Days:

sql

Copy code

```sql
WITH consecutive_logins AS (
    SELECT user_id, login_date,
           DATEDIFF(login_date, LAG(login_date) OVER (PARTITION BY user_id ORDER BY login_
    FROM logins
)
SELECT DISTINCT user_id
FROM consecutive_logins
WHERE day_diff = 1;
```

Career_W

7 Calculate Sales Ratio Between Two Categories:

```sql
WITH category_sales AS (
    SELECT category, SUM(sales) AS total_sales
    FROM sales
    GROUP BY category
)
SELECT c1.category, c2.category,
       (c1.total_sales * 1.0 / c2.total_sales) AS sales_ratio
FROM category_sales c1, category_sales c2
WHERE c1.category = 'Category A' AND c2.category = 'Category B';
```

8 Recursive Query for Hierarchical Structure:

```sql
WITH RECURSIVE employee_hierarchy AS (
    SELECT employee_id, manager_id, 1 AS level
    FROM employees
    WHERE manager_id IS NULL
    UNION ALL
    SELECT e.employee_id, e.manager_id, eh.level + 1
    FROM employees e
    INNER JOIN employee_hierarchy eh
    ON e.manager_id = eh.employee_id
)
SELECT * FROM employee_hierarchy;
```

Career

9️⃣ Find Gaps in Sequential Numbering:

```sql
SELECT current_value + 1 AS missing_value
FROM (
    SELECT column_name AS current_value,
           LEAD(column_name) OVER (ORDER BY column_name) AS next_value
    FROM table_name
) AS temp
WHERE next_value - current_value > 1;
```

🔟 Split Comma-Separated String into Rows:

```sql
SELECT value
FROM STRING_SPLIT('value1,value2,value3', ',');
```

💡 Advanced Problem-Solving:

1️⃣ Rank Products by Sales for Each Region:

```sql
SELECT region, product_id,
       RANK() OVER (PARTITION BY region ORDER BY sales DESC) AS rank
FROM product_sales;
```

2️⃣ Employees in the Top 10% of Salaries in Their Department:

```sql
SELECT emp_id, emp_name, salary
FROM employees
WHERE salary >= (
    SELECT PERCENTILE_CONT(0.9) WITHIN GROUP (ORDER BY salary)
    FROM employees
);
```

3️⃣ Orders Placed During Business Hours:

```sql
SELECT *
FROM orders
WHERE TIME(order_time) BETWEEN '09:00:00' AND '18:00:00';
```

4️⃣ Count Users Active on Both Weekdays and Weekends:

```sql
WITH user_logs AS (
    SELECT user_id,
           CASE
               WHEN DAYOFWEEK(login_date) IN (1, 7) THEN 'weekend'
               ELSE 'weekday'
           END AS day_type
    FROM logins
)
SELECT user_id
FROM user_logs
GROUP BY user_id
HAVING COUNT(DISTINCT day_type) = 2;
```

5️⃣ Customers Purchasing Across 3+ Categories:

```sql
SELECT customer_id
FROM sales
GROUP BY customer_id
HAVING COUNT(DISTINCT category) >= 3;
```

Career_w